REMARKS

Claims 1, 3, 5-16, 18-30, and 37-40 are pending in this application, of which claims 1, 14-16, and 37-40 are independent. Applicant respectfully submits that all of the pending claims are in condition for allowance. Applicant respectfully requests reconsideration of the outstanding rejections and allowance of all pending claims in view of the reasons set forth below.

I. Claim Rejection under 35 U.S.C. § 103(a)

In the Office Action, claims 1, 3, 5-16, 18-30, and 37-40 were rejected under 35 U.S.C. § 103(a) as being obvious over Conway ("Parsing with C++ deferred expressions," ACM SIGPLAN Notices, vol. 29, no. 9, pp. 9-16, ACM, 1994) (hereafter "Conway") in view of Walter et al. ("boost C++ Libraries uBLAS overview") (hereafter "Walter"). Applicant respectfully traverses the rejection.

A. Claim 1

Applicant's claim 1 recites:

1. In a program development environment, a computer implemented method comprising:
providing, via a programming language, a language processor with built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;
defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;
defining said assignment function in more than one class;
overloading said assignment function based on a context of said base language objects; and
calling said assignment function from said language processor to determine a value of at least one assignment within at least one of said base language and a base language extension to said base language.

Independent claim 1 recites the feature of ***overloading said assignment function based on a context of said base language objects***. Applicant respectfully submits that Conway and Walter, alone or in any reasonable combination, fail to disclose or suggest this feature of claim 1.

The Examiner recognizes that Conway does not disclose or suggest overloading an assignment function based on a context of base language objects (Office Action at page 3). The Examiner appears to offer contradictory interpretations of Conway. For example, although the Examiner states that the Evaluate() method of Conway performs "appropriate context evaluation," of the "base language parse tree objects," (Office Action at page 9), this interpretation contradicts the Examiner's statement on page 7 that "Conway does not explicitly disclose said evaluating done based on a context provided by said first parse tree objects." Applicants respectfully submit that the Examiner is correct in recognizing that Conway does not disclose overloading an assignment function based on a context of the base language objects, as stated by the Examiner at pages 3 and 7 of the Office Action.

As discussed in more detail below, Walter also does not disclose overloading an assignment function based on a context of base language objects. While Walter does utilize an assignment function, that assignment function is overloaded without regard to the context of the base language objects on which it operates.

Operator overloading is the situation in which an operator (such as "+" or "=") is implemented in a defined way based on the type of class involved in the operation. For example, in defining a new class, the class designer may overload the "+" operator so that the new class can be properly added in an equation. In claim 1, the context of the object, instead of only the type of class, is considered when overloading the assignment function.

The context of a base language object is described, for example, in the Specification at pages 2-3. In one example of overloading an operator based on context, an operator operating on an expression may assign the expression after taking into account the base language object's *kind* to determine if the object supports in-place operations. (Specification at page 8). Other examples of context are provided throughout the Specification. Overloading based on the context of the base language objects allows operations to be carried out more efficiently than in situations in which the operator is overloaded without regard for the context of the base language object, or with more user control.

For example, if an operator is overloaded without regard to the context of the base language object, the overloaded operator may fail to take into account that a particular operation may be performed in-place. In the example provided in the specification, the value "1" may be added to a matrix. Without context-based operator overloading, the conventional system creates a new temp matrix, adds one to each value in the matrix, and renaming the temp matrix as the original matrix. With the context-based operator overloading described in claim 1, an in-place operation can be performed whereby 1 is added to each element of the matrix without the need to create a temp matrix. This allows the operation to be performed more efficiently because additional memory need not be allocated for a temp variable.

The Examiner suggests that Walter discloses "evaluating the left hand side of the overloaded assignment operator" (Office Action at page 3). The Examiner states that "the Evaluate() method performs appropriate context evaluation for returning the left hand side value in the parser tree in different deferred assignment operations *by being redefined for new classes*" (Office Action at page 9). At this point, the Examiner appears to be referring to the Conway reference, as the Examiner cites "page 12" and the "Evaluate()" method, which appears to correspond to Conway and not Walter.

Although the Examiner appears to rely on Conway for "context evaluation," this contradicts the Examiner's statement at pages 3 and 7 of the Office Action. At pages 3 and 7, the Examiner correctly recognizes that Conway <u>does not disclose</u> overloading the assignment function based on the context of the base language object. The redefinition of the Evaluate() method for new classes is <u>not</u> an overloading of the assignment operator. Evaluation and assignment are <u>two different operations</u>. The Evaluate() method <u>evaluates</u> the value of a given expression. The Evaluate() method is not responsible for the <u>assignment function</u> and does not overload the assignment function based on a <u>context of the base language object</u>.

The Examiner appears to interpret the deferred assignment at page 12 of Conway as an overloading of the assignment operator based on the context of the base language objects. As noted above, this interpretation contradicts the Examiner's earlier interpretation that Conway <u>does not disclose</u> overloading an assignment operator based on the context of the base language objects. Further, a deferred assignment is <u>not</u> overloading an assignment operator based on a context of the base language object. A deferred assignment simply carries out an assignment at a later time, for example to allow the chaining or embedding of an assignment (Conway at page

12). Deferring an assignment does not overload the assignment operator based on the context of the base language object. Indeed, the same assignment is carried out in Conway as would have been carried out without the deferred assignment – it simply happens at a different time.

The Examiner also argues, at pages 8-9 of the Office Action, that Walter discloses ***overloading said assignment function based on a context of said base language objects*** at page 4, lines 1-7 (Office Action at page 9). Specifically, the Examiner argues that "Walter specifically teaches assigning the left hand side of the assignment operator from the evaluation of the right hand side of the assignment into a temporary and then assigning this temp value to the left hand side, *as in the instant Specification*" (Office Action at page 9).

However, the use of a temp variable is not the same as overloading the assignment function based on the context of the base language objects. Indeed, temp variables may be employed in the present Specification, but temp variables are also employed in conventional operator overloading (Specification at page 8). The use of a temp variable by itself in no way indicates that the operator is being overloaded ***based on a context of said base language objects.***

Indeed, in the example from the present Specification that the Examiner appears to be referencing (Specification at page 8), the use of temp variables to evaluate an expression is an example of the way that a *conventional operator overload* would be carried out, without regard to the context of the base language object. The cited example goes on to show that, by overloading the operator based on the context of the base language object, the operation may be carried out in place, without the use of temp variables. This represents an improvement because the temp objects need not be allocated at all, which saves memory and time.

For at least the reasons presented above, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest ***overloading said assignment function based on a context of said base language objects***, as present in independent claim 1. Claims 3 and 5-13 depend from claim 1 and, as such, include each and every element of claim 1. Therefore, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest each and every element of claims 3 and 5-13. Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claims 1, 3, and 5-13.

B. Claim 14

Applicant's independent claim 14 recites:

14. In a program development environment, a computer implemented method comprising:

providing, via a programming language, a language processor with built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including parse tree object methods that retrieve values for base language objects;

defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

defining said assignment function in more than one class;

overloading said assignment function based on a context of said base language objects, said context determined from one of said parse tree object methods;

calling said assignment function from said language processor to determine a value of at least one assignment within at least one of the said base language and a base language extension for said base language; and

generating code for an embedded processor using said parse tree data structure.

Independent claim 14 recites ***overloading said assignment function based on a context of said base language objects***. As noted above in the discussion of claim 1, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest ***overloading said assignment function based on a context of said base language objects.*** Therefore the combination of Conway and Walter do not disclose ***overloading said assignment function based on a context of said base language objects*** as recited in claim 14.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 14.

C. Claim 15

Applicant's independent claim 15 recites:

15. In a program development environment, a computer implemented method comprising:

providing, via a programming language, a language processor with built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;

defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

defining said assignment function in more than one class;

overloading said assignment function based on a context of said base language objects;

calling said assignment function from said language processor to determine a value of at least one assignment within at least one of the base language and a base language extension for said base language; and

using said parse tree data structure in software emulation.

Independent claim 15 recites ***overloading said assignment function based on a context of said base language***. As noted above in relation to claim 1, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above-quoted feature of claim 15.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 15.

D. <u>Claim 16 and 18-30</u>

Applicant's independent claim 16 recites:

16. A computer-readable storage medium for storing computer executable instructions for use in a program development environment, said instructions comprising:

one or more instructions for providing a programming language;

one or more instructions for providing a language processor via said programming language, said language processor having built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;

one or more instructions for defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

one or more instructions for defining said assignment function in more than one class;

one or more instructions for overloading said assignment function based on a context of said base language objects; and one or more instructions for calling said assignment function from said language processor to determine a value of at least one assignment within at least one of said base language and a base language extension to said base language.

Independent claim 16 recites ***one or more instructions for overloading said assignment function based on a context of said base language objects***. As noted above in relation to claim 1, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above-quoted feature of claim 16.

Conway and Walter, alone or in any reasonable combination, do not disclose or suggest each and every element of independent claim 16. Claims 18-30 depend from claim 16 and, as such, include each and every element of claim 16. Therefore, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest each and every element of claims 18-30. Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claims 16 and 18-30.

E. <u>Claim 37</u>

Applicant's independent claim 37 recites:

37. A computer-readable storage medium for storing computer executable instructions for use in a program development environment, said instructions comprising:

one or more instructions for providing a programming language;

one or more instructions for providing a language processor via said programming language, said language processor having built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;

one or more instructions for defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

one or more instructions for defining said assignment function in more than one class;

one or more instructions for overloading said assignment function based on a context of said base language objects;

one or more instructions for calling said assignment function from said language processor to determine a value of at least one assignment within at least one of said base language and a base language extension to said base language; and one or more instructions for generating code for an embedded processor using said parse tree data structure.

Independent claim 37 recites ***one or more instructions for overloading said assignment function based on a context of said base language objects***. As noted above in relation to claim 1, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above-quoted feature of claim 37.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 37.

F. Claim 38

Applicant's independent claim 38 recites:

38. A computer-readable storage medium for storing computer executable instructions for use in a program development environment, said instructions comprising:

one or more instructions for providing a programming language;

one or more instructions for providing a language processor via said programming language, said language processor having built-in support for a parse tree data structure written in a base language, said parse tree data structure represented as a class, said class being a basis for a plurality of parse tree objects, said parse tree objects including methods that retrieve values for base language objects;

one or more instructions for defining an assignment function, said assignment function taking a plurality of parse tree structures as arguments;

one or more instructions for defining said assignment function in more than one class;

one or more instructions for overloading said assignment function based on a context of said base language objects;

one or more instructions for calling said assignment function from said language processor to determine a value of at least one assignment within at least one of said base language and a base language extension to said base language; and

one or more instructions for using said parse tree data structure in software emulation.

Independent claim 38 recites ***one or more instructions for overloading said assignment function based on a context of said base language objects***. As noted above in relation to claim 1, Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above-quoted feature of claim 38.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 38.

G. <u>Claim 39</u>

Applicant's independent claim 39 recites:

> 39. In an object-oriented program development environment having a base language, a computer implemented method comprising:
> providing a programming language;
> providing a language processor via said programming language, said language processor having built-in support for a parse tree data structure in said object-oriented program development environment, said parse tree data structure used as a basis for at least one parse tree object, said parse tree objects including methods that retrieve values for base language objects;
> defining an assignment function taking a plurality of parse tree structures as arguments in more than one class;
> calling said assignment function from said language processor to determine a value of at least one assignment within at least one of a base language and a base language extension; and
> ***overloading a mathematical operator with said assignment function based on a context of a plurality of said base language objects.***

Independent claim 39 recites ***overloading a mathematical operator with said assignment function based on a context of a plurality of said base language objects***. Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above feature of claim 39.

The Examiner rejects claim 39 "for the same reasons set forth in connection with the rejection of claim 3 above" (Office Action at page 7). In rejecting claim 3, which depends from claim 1, the Examiner relies on the rejection of claim 1. In the rejection of claim 1, the Examiner recognizes that Conway does not disclose ***a context of a plurality of said base language objects***. As noted above with respect to claim 1, Walter also does disclose or suggest utilizing the <u>context</u> of the base language objects.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 39.

H. <u>Claim 40</u>

Applicant's independent claim 40 recites:

40. A computer implemented method for providing a language processor in a program development environment, said method comprising:

building a parse tree data structure based on source code with the language processor;

instantiating a first parse tree object and a second parse tree object, each parse tree object including methods that retrieve values for objects represented in the source code;

evaluating said second parse tree object to obtain a value, said evaluating done based on a context provided by said first parse tree object; and

assigning said value to said first parse tree object.

Independent claim 40 recites ***evaluating said second parse tree object to obtain a value, said evaluating done based on a context provided by said first parse tree object***. Conway and Walter, alone or in any reasonable combination, do not disclose or suggest the above feature of claim 40.

The Examiner recognizes that Conway does not disclose ***a context provided by said first parse tree object*** (Office Action at page 7). Walter also does disclose or suggest utilizing the <u>context</u> of an object. Further, Walter does not address parse trees, and so Walter does not disclose or suggest ***evaluating a second parse tree object to … <u>based on a context provided by said first parse tree object</u>***.

Therefore, Applicant respectfully requests that the Examiner reconsider and withdraw the 35 U.S.C. §103(a) rejection of claim 40.

CONCLUSION

In view of the above, Applicant believes the pending application is in condition for allowance and urges the Examiner to pass all of the pending claims to allowance. Should the Examiner feel that a teleconference would expedite the prosecution of this application, the Examiner is urged to contact the Applicant's attorney at (617) 227-7400.

Please charge any shortage or credit any overpayment of fees to our Deposit Account No. 12-0080, under Order No. MWS-039RCE. In the event that a petition for an extension of time is required to be submitted herewith, and the requisite petition does not accompany this response, the undersigned hereby petitions under 37 C.F.R. §1.136(a) for an extension of time for as many months as are required to render this submission timely. Any fee due is authorized to be charged to the aforementioned Deposit Account.

Dated: June 1, 2009

Respectfully submitted,

Electronic signature: /Kevin J. Canning/
Kevin J. Canning
Registration No.: 35,470
LAHIVE & COCKFIELD, LLP
One Post Office Square
Boston, Massachusetts 02109-2127
(617) 227-7400
(617) 742-4214 (Fax)
Attorney/Agent For Applicant